1) High-level approach — where to put security (the stages)

Implement DevSecOps across the full SDLC — shift-left + runtime controls:

1. **Plan** — threat modeling, policy baselines, security requirements (OWASP, CIS, SLSA).

2. **Code** — secure coding guidance, pre-commit hooks, SAST, secret scanning.

3. **Build** — dependency scanning, SBOM creation, reproducible builds, image scanning, sign artifacts.

4. **Test** — SAST, DAST, IAST (if used), infra tests (kics/checkov/tfsec), container runtime tests.

5. **Release** — artifact signing, registry policies, vulnerability gating, CVE policies.

6. **Deploy** — infra as code checks, policy enforcement (OPA/Gatekeeper), secure configs (network policies, PodSecurity).

7. **Operate / Observe** — runtime detection (Falco/Tetragon), EDR, logging, alerting, incident response.

8. **Governance** — compliance evidence, periodic audits, maturity metrics.

---

2) Tool map + where to implement + quick how-to examples

I'll show 1–2 concrete commands/config examples per area so you can copy-paste and adapt.

A. Linux host hardening (Plan → Operate)

Why: base of every server/container.
Tools: CIS Benchmarks, Lynis, Auditd, OS package manager updates, SELinux/AppArmor.

How to implement (example):

- Run CIS benchmark checklist and automate with scripts or Ansible.

- Example quick scan with **Lynis**:

```
sudo apt-get install lynis -y

sudo lynis audit system
```

- Enforce automatic security updates (example for Ubuntu):

```
sudo apt-get install unattended-upgrades

sudo dpkg-reconfigure --priority=low unattended-upgrades
```

- Enable Auditd and send logs to central SIEM (e.g., CloudWatch / Elasticsearch).

B. Docker image security & build-time (Build → Release)

Goals: minimal base images, vulnerability scanning, build reproducibility, image signing.

Tools: Dockerfile best practices, **Trivy**, **Clair/Anchore**, **Cosign (sigstore)**, Notary/Harbor.

How to implement:

- Best practice: use small, fixed base images (e.g., `python:3.11-slim`) and pin versions.

- CI step: scan image with **Trivy**:

```
# build image

docker build -t myapp:${GITHUB_SHA} .

# scan

trivy image --severity CRITICAL,HIGH myapp:${GITHUB_SHA}
```

- Create SBOM (with syft):

```
syft myapp:${GITHUB_SHA} -o json > sbom.json
```

- Sign image with **cosign**:

```
cosign sign --key cosign.key ghcr.io/org/myapp:${GITHUB_SHA}
```

- Registry policy: block images without signature or with high CVEs (Harbor / ECR image scanning + policy).

C. Source code security (Code stage)

Goals: find code vulnerabilities, secrets, insecure patterns early.

Tools: SonarQube, **Snyk**, **Semgrep**, **Checkmarx**, **GitGuardian**, **pre-commit** hooks.

How to implement:

- Add **pre-commit** + **detect-secrets**: `.pre-commit-config.yaml` (snippet):

```
repos:

- repo: https://github.com/Yelp/detect-secrets

  rev: v1.0.3

  hooks:

    - id: detect-secrets
```

- CI SAST with **Semgrep**:

```
semgrep --config=auto --output semgrep-results.json
```

- Add SonarQube or Snyk scanning step in CI and fail the build on threshold.

D. Dependency & OSS scanning (Build/Test)

Why: many vulnerabilities come from libs.

Tools: Snyk, **Dependabot**, **OWASP Dependency-Check**, **npm audit**, **Maven/Gradle scanners**.

How to implement:

- GitHub: enable Dependabot to auto-open PRs for vulnerable libraries.

- CI: run `snyk test` (or `mvn dependency-check:check` for Java).

```
snyk test --severity-threshold=high
```

E. Infrastructure as Code (Terraform) — shift-left (Plan → Test → Deploy)

Goals: detect misconfigs open ports, public S3, IAM privilege escalation before apply.

Tools: Checkov, **tfsec**, **Terrascan**, **OPA/Rego**.

How to implement:

- Local CI check:

```
# tfsec

tfsec ./terraform

# checkov

checkov -d ./terraform
```

- Example GitHub Action step to run tfsec and checkov — fail on issues.
- Enforce policy in pipeline and as pre-merge gates. Optionally use policy-as-code for PR review.

F. Cloud (AWS) secure configurations (Deploy → Operate)

Goals: least privilege IAM, logging, KMS/CMK, VPC isolations, S3 policies, ECR scanning, GuardDuty.

Tools: AWS Config rules, IAM Access Analyzer, AWS CloudTrail, GuardDuty, Security Hub, KMS, AWS Organizations SCPs, Inspector, ECR image scanning.

How to implement (examples):

- Enable CloudTrail organization-wide and send to secure S3 with lifecycle + encryption.
- Use AWS Config managed rules (e.g., `s3-bucket-public-read-prohibited`).
- Run **aws iam access-analyzer** and restrict overly-broad IAM policies.

- Automate checks (Checkov has AWS rules) and fail Terraform CI when public S3 or wide `0.0.0.0/0` ingress detected.

G. Kubernetes (K8s) — build & runtime controls (Deploy → Operate)

Goals: secure cluster admission, least-privilege RBAC, PodSecurity, network policies, image policies, secrets management, runtime detection.

Tools: Gatekeeper/OPA, **Kyverno**, **PodSecurityAdmission / PSP migration**, **Kube-bench** (CIS), **Trivy/Anchore** (image), **Falco / Tetragon / KubeArmor**, **kubesec**, **kube-hunter**, **kubescape**, **cert-manager**, **HashiCorp Vault / Kubernetes Secrets with KMS**.

How to implement (practical):

- Run **kube-bench** to check CIS controls:

```
kube-bench master
```

- Enforce admission policy with **Gatekeeper** example (deny privileged containers):

```
# Constraint template and Constraint to ban privileged containers
via Gatekeeper
```

(Deploy Gatekeeper / Kyverno and add constraints: `spec={containers[].securityContext.privileged: false}`)

- Enforce image policy: only allow signed images from trusted registry via **Cosign** verification + admission controller (e.g., **Kritis**, or custom OPA).

- Network policies: deny by default, create minimal allow-lists per namespace.

- Use **Falco** for runtime detection of suspicious syscalls (example rule: detect shell in a container).

- Secrets: store sensitive secrets in **Vault**, use CSI driver for K8s or external secret operators (External Secrets) to avoid K8s native secrets persistence.

H. CI/CD pipeline security (Build → Release)

Goals: protect pipeline credentials, least-privilege runners, immutable builds, artifact promotion.

Tools: GitHub Actions, GitLab CI, Jenkins with Role-based access, HashiCorp Vault, OIDC-based short-lived credentials, signed artifacts.

How to implement (examples):

- Use OIDC from GitHub Actions to obtain short-lived AWS creds (no long-lived secrets).

- Protect branches, require code review, and enable 2FA for repos.

- Example GitHub Action snippet (runs tests, SAST, container scan):

```
name: CI

on: [push, pull_request]

jobs:

  build:
```

```
    runs-on: ubuntu-latest

    steps:

      - uses: actions/checkout@v4

      - name: Install Trivy

        run: sudo apt-get install -y trivy

      - name: Build container

        run: docker build -t myapp:${{ github.sha }} .

      - name: SAST Semgrep

        run: semgrep --config=auto

      - name: Image scan

        run: trivy image --severity HIGH,CRITICAL myapp:${{
github.sha }}

      - name: Terraform check

        run: |

          curl -sSL
https://raw.githubusercontent.com/bridgecrewio/checkov/master/instal
l.sh | bash

          checkov -d ./terraform
```

- Use least-privilege service accounts for runners and vault secrets injection.

I. Secrets management

Tools: HashiCorp Vault, AWS Secrets Manager, Azure Key Vault, Sealed Secrets, External Secrets operator.

How to implement:

- Put secrets in Vault, enable dynamic secrets for DBs. Inject into CI/CD using short-lived tokens (Vault Agent, Kubernetes CSI driver).

- Avoid committing secrets — enforce with pre-commit & secret scanning.

J. Runtime protection & detection (Operate)

Goals: detect anomalies, commands, privilege escalation, suspicious network flows.

Tools: Falco, **Tetragon**, **KubeArmor**, EDR solutions, SIEM (Splunk/ELK/Grafana Cloud), Prometheus + Alertmanager.

How to implement:

- Deploy **Falco** as DaemonSet with custom rules and send alerts to Slack/SIEM.

- Example Falco rule snippet (detect shell inside a container):

```
- rule: Shell In Container

  desc: Detect shell execution inside container

  condition: container and evt.type = execve and proc.name in (bash, sh, python)
```

```
  output: "Shell in container (user=%user.name command=%proc.cmdline
container=%container.id image=%container.image)"

  priority: WARNING
```

- Correlate logs with Prometheus metrics and alerts.

K. Supply-chain & artifact trust

Concepts/Tools: SBOM (Syft), **SLSA** principles, **cosign** (sigstore), reproducible builds.

How to implement:

- Generate SBOM at build (`syft`), store with release artifacts.
- Sign artifacts/images with `cosign` and enforce verification before deploy.

L. Secrets / credentials hygiene in CI/CD

Practices: OIDC, ephemeral credentials, no hard-coded secrets, vault.

M. Compliance & standards mapping

Standards to align with:

- **CIS Benchmarks (Linux, K8s)** — host & cluster hardening.

- **OWASP Top 10 / ASVS** — web app vulnerabilities, SAST/DAST tests.
- **NIST SP 800-53 / 800-190** — cloud & container controls.
- **ISO 27001** — InfoSec management.
- **SLSA** — secure software supply chain.
- **PCI-DSS / SOC2** — if handling card data or customer trust.

Implementation mapping example:

- CIS K8s -> use kube-bench, Gatekeeper policies.
- OWASP -> SAST (Semgrep/Sonar) + DAST (ZAP) in test stage.
- SLSA -> sign artifacts, generate SBOM, build provenance.

---

3) Prioritized practical checklist (quick wins → mid-term → long-term)

0–30 days (Quick wins)

- Enable repo branch protection and 2FA.
- Add pre-commit hooks for secrets scanning.

- Add Trivy scanning step for images in CI.
- Run tfsec/checkov on Terraform locally and in CI.
- Enable CloudTrail/Cloud logging and basic GuardDuty/Inspector.

30–90 days (Solidify)

- Implement OIDC for CI -> AWS (remove long-lived creds).
- Deploy Gatekeeper/Kyverno with 2–4 important constraints (no privileged containers, disallow hostPath).
- Integrate SAST (Semgrep/Sonar) and dependency scanning (Snyk).
- Deploy Falco for runtime alerts; centralize logs.

90+ days (Mature)

- Full policy-as-code for infra and images, signed artifacts, SBOMs, SLSA alignment.
- Automated incident response playbooks + tabletop exercises.
- Continuous compliance reporting dashboards (Security Hub / Cloud Security Posture).

---

4) Example quick pipeline checklist (CI gates)

- On PR: run `pre-commit`, Semgrep, Dependency scan.

- On merge to main: run build -> generate SBOM -> image scan (Trivy) -> sign image -> terraform plan -> checkov

- Deploy jobs require signed artifact + OPA policy pass.

---

5) Helpful commands & snippets (copy-paste)

- Trivy image scan:

```
trivy image --severity HIGH,CRITICAL --exit-code 1
myapp:${GITHUB_SHA}
```

- Checkov (Terraform):

```
checkov -d ./terraform --compact
```

- tfsec:

```
tfsec .
```

- Semgrep:

```
semgrep --config=p/ci
```

- Cosign sign:

```
cosign sign --key cosign.key ghcr.io/org/myapp:${GITHUB_SHA}
```

- Generate SBOM:

```
syft myapp:${GITHUB_SHA} -o json > sbom.json
```

---

6) Metrics to track (what proves progress)

- % of images scanned & passing policy
- % of Terraform plans run through checkov before apply
- Mean time to detect (MTTD) / Mean time to remediate (MTTR) for vulnerabilities
- % of PRs with SAST passing before merge
- Number of secrets found in git per month (should drop to 0)

---

7) Quick risk-based priority (if you have limited time)

1. Prevent secrets and leaked creds.

2. Image scanning & signing.

3. IaC policy checks (Terraform).

4. Admission policies in K8s (no privileged containers, signed images).

5. Runtime detection (Falco) + centralized logging.

1. Major Compliance Standards & Certifications

Global

- **ISO/IEC 27001** — Information Security Management System (ISMS).

- **ISO/IEC 27017** — Cloud security.

- **ISO/IEC 27018** — Privacy in cloud (PII).

- **SOC 2 (Type I/II)** — Service provider trust principles (security, availability, confidentiality, processing integrity, privacy).

- **PCI DSS** — Payment card industry (credit card data).

- **NIST Cybersecurity Framework (CSF)** — Identify, Protect, Detect, Respond, Recover.

- **NIST 800-53 / FedRAMP** — U.S. government cloud security baseline.

- **SLSA (Supply-chain Levels for Software Artifacts)** — Secure software supply chain.

US-specific

- **HIPAA** — Health data protection (PHI).

- **HITECH** — Health tech security.

- **CMMC** — U.S. defense contractors.

- **FISMA** — Federal agency security.

EU / UK

- **GDPR** — Data protection & privacy.

- **UK Cyber Essentials** — Basic IT security certification.

- **DORA** — EU Digital Operational Resilience Act (for financial services).

APAC

- **India – DPDPA 2023** (Digital Personal Data Protection Act).

- **Singapore – PDPA**.

- **Australia – Essential 8 / ISM**.

- **Japan – APPI (Act on Protection of Personal Information)**.

---

2. What Each Requires (Themes)

1. Access Control (IAM, RBAC)

Every framework requires you to control *who* can access *what*.

- Example: Developers shouldn't have production DB admin rights.
- Implementation: IAM roles, MFA, RBAC in Kubernetes, least-privilege policies.
  ✅ Required in **all frameworks**.

---

2. Encryption (Data at Rest & in Transit)

Data must be protected both when stored and when sent over networks.

- Example: Encrypt EBS volumes with KMS, use TLS for APIs.
- Evidence: Terraform configs, Cloud provider settings.
  ✅ Required in **all frameworks**.

---

3. Vulnerability Management

Organizations must identify, track, and fix software/hardware vulnerabilities.

- Example: Run Trivy scans on Docker images, patch Linux hosts, scan Terraform with Checkov.
- GDPR is lighter here (focuses more on privacy), but others require structured vulnerability management.
  ✅ Strongly required in **ISO, SOC 2, PCI DSS, HIPAA, NIST**; partial in GDPR.

---

4. Change Management (CI/CD Evidence)

Every system change must be approved, tested, and documented.

- Example: GitHub PR approvals, CI/CD pipeline logs, Terraform plan approvals.
- Evidence: Git logs, Jenkins/GitHub Actions history.
  ✅ Required in most standards (ISO, SOC 2, PCI DSS, HIPAA, NIST). GDPR doesn't enforce, but benefits from it.

---

5. Logging & Monitoring

You must log all important activities and monitor for threats.

- Example: AWS CloudTrail, ELK stack, Falco alerts, SIEM dashboards.
- Evidence: Centralized log storage + retention policy.
  ✅ Required in **all frameworks**.

---

6. Incident Response Plan

You must have a documented process to detect, respond, and recover from incidents.

- Example: PagerDuty alerts, runbooks, post-mortems.
- Evidence: Incident response policy, drill results.
  ✅ Required in **all frameworks**.

---

7. Data Privacy & Retention

Defines how you collect, store, use, and delete sensitive data.

- Example: GDPR requires you to delete user data on request. PCI DSS requires protecting cardholder data. HIPAA requires strict PHI handling.
- Evidence: Data retention policy, anonymization/masking in logs.
  ✅ Strong in **GDPR, HIPAA, PCI DSS**, lighter in ISO, SOC 2, NIST.

---

8. Third-party Risk Management

You must ensure vendors and partners also comply with security standards.

- Example: Cloud providers must provide SOC 2/ISO reports, you run vendor risk assessments.
- Evidence: Vendor security questionnaires, signed compliance certificates.
  ✅ Strong in **ISO, SOC 2, HIPAA, GDPR, NIST**. PCI DSS is weaker here.

---

9. Secure SDLC / DevSecOps Evidence

You need proof that software is developed securely from design → code → deploy.

- Example: SAST scans, IaC checks, signed Docker images, PR approvals, threat modeling.
- Evidence: CI/CD logs, SAST/DAST reports, SBOMs.
  ✅ Required in **SOC 2, PCI DSS, HIPAA, GDPR, NIST**, recommended in ISO.

---

3. How DevSecOps Engineers Contribute to Compliance

👉 Compliance officers write policies, but **engineers provide the technical controls + evidence**.
Here's how we tie our work to governance:

Identity & Access Management (IAM)

- Enforce SSO/MFA (Okta, AWS SSO).
- Use least-privilege IAM roles.
  Evidence: IAM policy docs, access review logs, AWS Access Analyzer reports.

Encryption

- Enforce TLS everywhere.

- Use KMS (AWS/GCP/Azure) for at-rest encryption.
  Evidence: Cloud config screenshots, Terraform with `server_side_encryption = "AES256"`.

Vulnerability Management

- Automated scans: Trivy, Snyk, Checkov, tfsec.
  Evidence: CI/CD reports stored in artifact repo, monthly scan summaries.

Change Management

- PR approvals, Git history, pipeline logs.
  Evidence: GitHub/GitLab audit logs, merge request history, Jenkins job history.

Logging & Monitoring

- Centralize with ELK/CloudWatch/Datadog.

- Alerting with Prometheus + Alertmanager.
  Evidence: SIEM dashboards, alert response logs, retention policies.

Incident Response

- Define playbooks (pager duty, runbooks).
  Evidence: Runbook document, incident post-mortems, proof of tabletop exercises.

Data Privacy & Retention

- Mask PII in logs.

- Implement retention policies (delete logs after X days).
  Evidence: Log lifecycle configs, anonymization scripts, database data retention configs.

Secure SDLC (DevSecOps)

- Pre-commit hooks, SAST, DAST, IaC scanning, policy-as-code.
  Evidence: CI/CD pipeline YAML files, scan reports, admission controller logs.

Continuous Compliance Dashboards

- Tools: **AWS Security Hub**, **Cloud Custodian**, **Prisma Cloud**, **Aqua Security**, **Kubescape**, **DefectDojo**.
  Evidence: Exported compliance posture dashboards mapped to ISO/SOC2/PCI DSS.

---

4. Evidence Examples (What Auditors Expect)

ISO 27001

- Risk assessment document.
- IAM access review logs.
- Vulnerability scan reports.
- Change control evidence (GitHub PR approvals).

SOC 2

- Evidence of CI/CD controls.
- CloudTrail logs showing admin access.
- Monthly vulnerability scan results.
- Monitoring dashboards proving uptime (availability).

PCI DSS

- Firewall rules (Terraform code).
- Encrypted DB config (RDS encryption enabled).
- Segregated VPC evidence.
- Quarterly ASV (Approved Scan Vendor) reports.

HIPAA

- Audit trail of PHI access.
- Evidence of encryption (EBS volumes with KMS).
- Incident response drill documents.

GDPR / DPDPA

- Data retention policies.
- Proof of data deletion on user request.
- Encryption evidence (TLS certs, key mgmt).

FedRAMP / NIST 800-53

- Control implementation matrix (NIST -> technical controls).
- IAM least privilege + logging configs.
- Continuous monitoring reports.

---

5. DevSecOps Evidence Mapping (Cheat Sheet)

DevOps Practice	Compliance Evidence Produced

Trivy image scan in CI	PCI DSS (vuln mgmt), ISO 27001 (control A.12.6.1)
Terraform + Checkov	ISO 27001 (A.14.2.8 Secure Dev)
Signed images (cosign)	SLSA, SOC 2, NIST (supply chain integrity)
Kubernetes Gatekeeper policies	ISO 27001 (least privilege), SOC 2 (access control)
Falco runtime detection	NIST (IR-5), ISO (logging/monitoring)
Vault secret rotation	ISO (A.9 Access Control), SOC 2 (confidentiality)
CloudTrail enabled	SOC 2, ISO (A.12.4 Event Logging), PCI DSS
PR approvals in GitHub	SOC 2 (change management), ISO 27001
GuardDuty enabled	FedRAMP, NIST 800-53 (RA-5, SI-4)

---

6. Path to Achieve Certifications (Engineering POV)

Step 1 — Baseline controls

- Apply CIS Benchmarks (Linux, AWS, K8s).
- Central logging & monitoring.
- Enable encryption everywhere.

Step 2 — Map controls to frameworks

- Use compliance mapping tools (Prisma, AWS Audit Manager, Cloud Custodian).
- Maintain a **control matrix**: requirement → implementation → evidence.

Step 3 — Automate evidence collection

- Export CI/CD scan reports to S3 bucket.
- Forward CloudTrail/Audit logs to SIEM.
- Archive monthly compliance posture dashboards.

Step 4 — Run internal audits

- Mock audits every quarter.
- Generate compliance reports for ISO/SOC2 (tools: AWS Audit Manager, Drata, Vanta).

Step 5 — External certification

- Engage auditor, provide evidence docs, walkthrough demos.

- Maintain continuous compliance dashboards for real-time evidence.

Tooling

Pre-commit time tools

In this section you can find lifecycle helpers, precommit hook tools and threat modeling tools. Threat modeling tools are specific category by themselves allowing you to simulate and discover potential gaps before you start to develop the software or during the process.

Modern DevSecOps tools allow using Threat modeling as code or generation of threat models based on the existing code annotations.

Name	URL	Description	Meta
git-secrets	https://github.com/awslabs/git-secrets	AWS labs tool preventing you from committing secrets to a git repository	⚠
git-hound	https://github.com/tillson/git-hound	Searchers secrets in git	⚠
goSDL	https://github.com/slackhq/goSDL	Security Development Lifecycle checklist	⚠
ThreatPlaybook	https://github.com/we45/ThreatPlaybook	Threat modeling as code	⚠

Threat Dragon	https://github.com/OWASP/threat-dragon	OWASP Threat modeling tool	⚠
threatspec	https://github.com/threatspec/threatspec	Threat modeling as code	⚠
pytm	https://github.com/izar/pytm	A Pythonic framework for threat modeling	⚠
Threagile	https://github.com/Threagile/threagile	A Go framework for threat modeling	⚠
MAL-lang	https://mal-lang.org/#what	A language to create cyber threat modeling systems for specific domains	⚠
Microsoft Threat modeling tool	https://docs.microsoft.com/en-us/azure/security/develop/threat-modeling-tool	Microsoft threat modeling tool	⚠
Talisman	https://github.com/thoughtworks/talisman	A tool to detect and prevent secrets from getting checked in	⚠

SEDATED	https://github.com/OWASP/SEDATED	The SEDATED ® Project (Sensitive Enterprise Data Analyzer To Eliminate Disclosure) focuses on preventing sensitive data such as user credentials and tokens from being pushed to Git.	⚠
Sonarlint	https://github.com/SonarSource/sonarlint-core	Sonar linting utility for IDE	⚠
DevSkim	https://github.com/microsoft/DevSkim	DevSkim is a framework of IDE extensions and language analyzers that provide inline security analysis	⚠
detect-secrets	https://github.com/Yelp/detect-secrets	Detects secrets in your codebase	⚠

tflint	https://github.com/terraform-linters/tflint	A Pluggable Terraform Linter	⚠
Steampipe Code Plugin	https://github.com/turbot/steampipe-plugin-code	Use SQL to detect secrets from source code and data sources.	⚠

Secrets management

Secrets management includes managing, versioning, encryption, discovery, rotating, provisioning of passwords, certificates, configuration values and other types of secrets.

Name	URL	Description	Meta
GitLeaks	https://github.com/zricethezav/gitleaks	Gitleaks is a scanning tool for detecting hardcoded secrets	⚠
ggshield	https://github.com/gitguardian/ggshield	GitGuardian shield (ggshield) is a CLI application that runs in your local environment or in a CI environment and helps you detect more than 350+ types of secrets and sensitive files.	⚠
TruffleHog	https://github.com/trufflesecurity/trufflehog	TruffleHog is a scanning tool for detecting hardcoded secrets	⚠

Hashicorp Vault	https://github.com/hashicorp/vault	Hashicorp Vault secrets management	⚠
Mozilla SOPS	https://github.com/mozilla/sops	Mozilla Secrets Operations	⚠
AWS secrets manager GH action	https://github.com/marketplace/actions/aws-secrets-manager-actions	AWS secrets manager docs	⚠
GitRob	https://github.com/michenriksen/gitrob	Gitrob is a tool to help find potentially sensitive files pushed to public repositories on Github	⚠
git-wild-hunt	https://github.com/d1vious/git-wild-hunt	A tool to hunt for credentials in the GitHub	⚠
aws-vault	https://github.com/99designs/aws-vault	AWS Vault is a tool to securely store and access AWS credentials in a development environment	⚠
Knox	https://github.com/pinterest/knox	Knox is a service for storing and rotation of secrets, keys, and passwords used by other services	⚠

Chef vault	https://github.com/chef/chef-vault	allows you to encrypt a Chef Data Bag Item	⚠
Ansible vault	Ansible vault docs	Encryption/decryption utility for Ansible data files	⚠

OSS and Dependency management

Dependency security testing and analysis is very important part of discovering supply chain attacks. SBOM creation and following dependency scanning (Software composition analysis) is critical part of continuous integration (CI). Data series and data trends tracking should be part of CI tooling. You need to know what you produce and what you consume in context of libraries and packages.

Name	URL	Description	Meta
CycloneDX	https://github.com/orgs/CycloneDX/repositories	CycloneDX format for SBOM	⚠
cdxgen	https://github.com/AppThreat/cdxgen	Generates CycloneDX SBOM, supports many languages and package managers.	⚠
SPDX	https://github.com/spdx/spdx-spec	SPDX format for SBOM - Software Package Data Exchange	⚠

Snyk	https://github.com/snyk/snyk	Snyk scans and monitors your projects for security vulnerabilitie s	
vulncost	https://github.com/snyk/vulncost	Security Scanner for VS Code	
Dependency Combobulator	https://github.com/apiiro/combobulator	Dependency-related attacks detection and prevention through heuristics and insight engine (support multiple dependency schemes)	
DependencyTrack	https://github.com/DependencyTrack/dependency-track	Dependency security tracking platform	
DependencyCheck	https://github.com/jeremylong/DependencyCheck	Simple dependency security scanner good for CI	
Retire.js	https://github.com/retirejs/retire.js/	Helps developers to detect the use of JS-library versions with known	

		vulnerabilities	
PHP security checker	https://github.com/fabpot/local-php-security-checker	Check vulnerabilities in PHP dependencies	⚠
bundler-audit	https://github.com/rubysec/bundler-audit	Patch-level verification for bundler	⚠
gemnasium	https://gitlab.com/gitlab-org/security-products/analyzers/gemnasium	Dependency Scanning Analyzer based on Gemnasium	
Dependabot	https://github.com/dependabot/dependabot-core	Automated dependency updates built into GitHub providing security alerts	⚠
Renovatebot	https://github.com/renovatebot/renovate	Automated dependency updates, patches multi-platform and multi-language	⚠
npm-check	https://www.npmjs.com/package/npm-check	Check for outdated, incorrect, and unused dependencies.	⚠

Security Scorecards	https://securityscorecards.dev	Checks for several security health metrics on open source libraries and provides a score (0-10) to be considered in the decision making of what libraries to use.	⚠
Syft	https://github.com/anchore/syft	CLI tool and library for generating an SBOM from container images (and filesystems).	⚠
OSS Review Toolkit	https://github.com/oss-review-toolkit/ort	A suite of tools to automate software compliance checks.	⚠

Supply chain specific tools

Supply chain is often the target of attacks. Which libraries you use can have a massive impact on security of the final product (artifacts). CI (continuous integration) must be monitored inside the tasks and jobs in pipeline steps. Integrity checks must be stored out of the system and in ideal case several validation runs with comparison of integrity hashes / or attestation must be performed.

Name	URL	Description	Meta

Tekton chains	https://github.com/tektoncd/chains	Kubernetes Custom Resource Definition (CRD) controller that allows you to manage your supply chain security in Tekton.	⚠
in-toto	https://github.com/in-toto/attestation/tree/v0.1.0/spec	An in-toto attestation is authenticated metadata about one or more software artifacts	⚠
SLSA	Official GitHub link	Supply-chain Levels for Software Artifacts	⚠
kritis	https://github.com/grafeas/kritis	Solution for securing your software supply chain for Kubernetes apps	⚠
ratify	https://github.com/deislabs/ratify	Artifact Ratification Framework	⚠
chain-bench	https://github.com/aquasecurity/chain-bench	Supply Chain Audit Tool	⚠

SAST

Static code review tools working with source code and looking for known patterns and relationships of methods, variables, classes and libraries. SAST works with the raw code and usually not with build packages.

Name	URL	Description	Meta
Brakeman	https://github.com/presidentbeef/brakeman	Brakeman is a static analysis tool which checks Ruby on Rails applications for security vulnerabilities	
Semgrep	https://semgrep.dev/	Hi-Quality Open source, works on 17+ languages	
Bandit	https://github.com/PyCQA/bandit	Python specific SAST tool	
libsast	https://github.com/ajinabraham/libsast	Generic SAST for Security Engineers. Powered by regex based pattern matcher and semantic aware semgrep	
ESLint	https://eslint.org/	Find and fix problems in your JavaScript code	
nodejsscan	https://github.com/ajinabraham/nodejsscan	NodeJs SAST scanner with GUI	

FindSecurity Bugs	https://find-sec-bugs.github.io/	The SpotBugs plugin for security audits of Java web applications	
SonarQube community	https://github.com/SonarSource/sonarqube	Detect security issues in code review with Static Application Security Testing (SAST)	
gosec	https://github.com/securego/gosec	Inspects source code for security problems by scanning the Go AST.	
Safety	https://github.com/pyupio/safety	Checks Python dependencies for known security vulnerabilities .	
Bearer	https://github.com/Bearer/bearer	Detect security issues in various languages (JavaScript/Type Script, Ruby, Java, PHP...) .	
mobsfscan	https://github.com/MobSF/mobsfscan	Detect security issues in Android and iOS source code (Java/Kotlin and Objective C/Swift)	

Note: Semgrep is free CLI tool, however some rulesets (https://semgrep.dev/r) are having various licences, some can be free to use and can be commercial.

OWASP curated list of SAST tools :
https://owasp.org/www-community/Source_Code_Analysis_Tools

DAST

Dynamic application security testing (DAST) is a type of application testing (in most cases web) that checks your application from the outside by active communication and analysis of the responses based on injected inputs. DAST tools rely on inputs and outputs to operate. A DAST tool uses these to check for security problems while the software is actually running and is actively deployed on the server (or serverless function).

Name	URL	Description	Meta
Zap proxy	https://owasp.org/www-project-zap/	Zap proxy providing various docker containers for CI/CD pipeline	
Akto	https://github.com/akto-api-security/akto/)	API Security Testing with 150+ YAML Tests	
Wapiti	https://github.com/wapiti-scanner/wapiti	Light pipeline ready scanning tool	
Nuclei	https://github.com/projectdiscovery/nuclei	Template based security scanning tool	
purpleteam	https://github.com/purpleteam-labs/purpleteam	CLI DAST tool incubator project	

oss-fuzz	https://github.com/google/oss-fuzz	OSS-Fuzz: Continuous Fuzzing for Open Source Software	⚠
nikto	https://github.com/sullo/nikto	Nikto web server scanner	⚠
skipfish	https://code.google.com/archive/p/skipfish/	Skipfish is an active web application security reconnaissance tool	⚠

IAST

Name	URL	Description	Meta
CakeFuzzer	https://github.com/Zigrin-Security/CakeFuzzer	Cake Fuzzer is a project that is meant to help automatically and continuously discover vulnerabilities in CakePHP based web applications with very limited false positives.	⚠

Continuous deployment security

Name	URL	Description	Meta

SecureCode Box	https://github.com/secureCodeBox/secureCodeBox	Toolchain for continuous scanning of applications and infrastructure	⚠
OpenSCAP	https://github.com/OpenSCAP/openscap	Open Source Security Compliance Solution	⚠
ThreatMapper	https://github.com/deepfence/ThreatMapper	ThreatMapper hunts for vulnerabilities in your production platforms, and ranks these vulnerabilities based on their risk-of-exploit.	⚠

Kubernetes

Name	URL	Description	Meta
KubiScan	https://github.com/cyberark/KubiScan	A tool for scanning Kubernetes cluster for risky permissions	⚠

Kubeaudit	https://github.com/Shopify/kubeaudit	Audit Kubernetes clusters for various different security concerns	⚠
Kubescape	https://github.com/armosec/kubescape	The first open-source tool for testing if Kubernetes is deployed according to the NSA-CISA and the MITRE ATT&CK®.	⚠
kubesec	https://github.com/controlplaneio/kubesec	Security risk analysis for Kubernetes resources	⚠
kube-bench	https://github.com/aquasecurity/kube-bench	Kubernetes benchmarking tool	⚠
kube-score	https://github.com/zegl/kube-score	Static code analysis of your Kubernetes object definitions	⚠
kube-hunter	https://github.com/aquasecurity/kube-hunter	Active scanner for k8s (purple)	⚠

Calico	https://github.com/projectcalico/calico	Calico is an open source networking and network security solution for containers	⚠
Krane	https://github.com/appvia/krane	Simple Kubernetes RBAC static analysis tool	⚠
Gatekeeper	https://github.com/open-policy-agent/gatekeeper	Open policy agent gatekeeper for k8s	⚠
Inspektor-gadget	https://github.com/kinvolk/inspektor-gadget	Collection of tools (or gadgets) to debug and inspect k8s	⚠
kube-linter	https://github.com/stackrox/kube-linter	Static analysis for Kubernetes	⚠
mizu-api-traffic-viewer	https://github.com/up9inc/mizu	A simple-yet-powerful API traffic viewer for Kubernetes enabling you to view all API communication between microservices to help your debug and	⚠

		troubleshoot regressions.	
HelmSnyk	https://github.com/snyk-labs/helm-snyk	The Helm plugin for Snyk provides a subcomman d for testing the images.	⚠
Kubewarden	https://github.com/orgs/kubewarden/repos itories	Policy as code for kubernetes from SUSE.	⚠
Kubernetes-si gs BOM	https://github.com/kubernetes-sigs/bom	Kubernetes BOM generator	⚠
Capsule	https://github.com/clastix/capsule	A multi-tenanc y and policy-based framework for Kubernetes	⚠
Badrobot	https://github.com/controlplaneio/badrobot	Badrobot is a Kubernetes Operator audit tool	⚠
kube-scan	https://github.com/octarinesec/kube-scan	k8s cluster risk assessment tool	⚠

Istio	https://istio.io	Istio is a service mesh based on Envoy. Engage encryption, role-based access, and authentication across services.	
Kubernetes Insights	https://github.com/turbot/steampipe-mod-kubernetes-insights	Visualize Kubernetes inventory and permissions through relationship graphs.	
Kubernetes Compliance	https://github.com/turbot/steampipe-mod-kubernetes-compliance	Check compliance of Kubernetes configurations to security best practices.	
trivy-operator	https://github.com/aquasecurity/trivy-operator	Kubernetes-native security toolkit.	

Containers

Name	URL	Description	Meta

Harbor	https://github.com/goharbor/harbor	Trusted cloud native registry project	
Anchore	https://github.com/anchore/anchore-engine	Centralized service for inspection, analysis, and certification of container images	
Clair	https://github.com/quay/clair	Docker vulnerability scanner	
Deepfence ThreatMapper	https://github.com/deepfence/ThreatMapper	Apache v2, powerful runtime vulnerability scanner for kubernetes, virtual machines and serverless.	
Docker bench	https://github.com/docker/docker-bench-security	Docker benchmarking against CIS	
Falco	https://github.com/falcosecurity/falco	Container runtime protection	
Trivy	https://github.com/aquasecurity/trivy	Comprehensive scanner for vulnerabilities in	

		container images	
Notary	https://github.com/notaryproject/notary	Docker signing	
Cosign	https://github.com/sigstore/cosign	Container signing	
watchtower	https://github.com/containrrr/watchtower	Updates the running version of your containerized app	
Grype	https://github.com/anchore/grype	Vulnerability scanner for container images (and also filesystems).	
Copacetic	https://github.com/project-copacetic/copacetic	CLI tool for directly patching container images	

Multi-Cloud

Name	URL	Description	Meta

Cloudsploit	https://github.com/aquasecurity/cloudsploit	Detection of security risks in cloud infrastructure	
ScoutSuite	https://github.com/nccgroup/ScoutSuite	NCCgroup mutlicloud scanning tool	
CloudCustodian	https://github.com/cloud-custodian/cloud-custodian/	Multicloud security analysis framework	
CloudGraph	https://github.com/cloudgraphdev/cli	GraphQL API + Security for AWS, Azure, GCP, and K8s	
Steampipe	https://github.com/turbot/steampipe	Instantly query your cloud, code, logs & more with SQL. Build on thousands of open-source benchmarks & dashboards for security & insights.	

AWS

AWS specific DevSecOps tooling. Tools here cover different areas like inventory management, misconfiguration scanning or IAM roles and policies review.

Name	URL	Description	Meta
Prowler	https://github.com/toniblyx/prowler	Prowler is a command line tool that helps with AWS security assessment, auditing, hardening and incident response.	
aws-inventory	https://github.com/nccgroup/aws-inventory	Helps to discover all AWS resources created in an account	
PacBot	https://github.com/tmobile/pacbot	Policy as Code Bot (PacBot)	
Komiser	https://github.com/mlabouardy/komiser	Monitoring dashboard for costs and security	
Cloudsplaining	https://github.com/salesforce/cloudsplaining	IAM analysis framework	
ElectricEye	https://github.com/jonrau1/ElectricEye	Continuously monitor your AWS services for configurations	

Cloudmapper	https://github.com/duo-labs/cloudmapper	CloudMapper helps you analyze your Amazon Web Services (AWS) environments	
cartography	https://github.com/lyft/cartography	Consolidates AWS infrastructure assets and the relationships between them in an intuitive graph	
policy_sentry	https://github.com/salesforce/policy_sentry	IAM Least Privilege Policy Generator	
AirIAM	https://github.com/bridgecrewio/AirIAM	IAM Least Privilege anmalyzer and Terraformer	
StreamAlert	https://github.com/airbnb/streamalert	AirBnB serverless, real-time data analysis framework which empowers you to ingest, analyze, and alert	

CloudQuery	https://github.com/cloudquery/cloudquery/	AirBnB serverless, real-time data analysis framework which empowers you to ingest, analyze, and alert	
S3Scanner	https://github.com/sa7mon/S3Scanner/	A tool to find open S3 buckets and dump their contents	
aws-iam-authenticator	https://github.com/kubernetes-sigs/aws-iam-authenticator/	A tool to use AWS IAM credentials to authenticate to a Kubernetes cluster	
kube2iam	https://github.com/jtblin/kube2iam/	A tool to use AWS IAM credentials to authenticate to a Kubernetes cluster	
AWS open source security samples	Official AWS opensource repo	Collection of official AWS open-source resources	
AWS Firewall factory	Globaldatanet FMS automation	Deploy, update, and stage your WAFs while managing	

		them centrally via FMS	
Parliment	Parliment	Parliament is an AWS IAM linting library	
Yor	Yor	Adds informative and consistent tags across infrastructure-as-code frameworks such as Terraform, CloudFormation, and Serverless	
AWS Insights	https://github.com/turbot/steampipe-mod-aws-insights	Visualize AWS inventory and permissions through relationship graphs.	
AWS Compliance	https://github.com/turbot/steampipe-mod-aws-compliance	Check compliance of AWS configurations to security best practices.	

Google cloud platform

GCP specific DevSecOps tooling. Tools here cover different areas like inventory management, misconfiguration scanning or IAM roles and policies review.

Name	URL	Description	Meta
Forseti	https://github.com/forseti-security/forseti-security	Complex security orchestration and scanning platform	⚠
GCP Insights	https://github.com/turbot/steampipe-mod-gcp-insights	Visualize GCP inventory and permissions through relationship graphs.	⚠
GCP Compliance	https://github.com/turbot/steampipe-mod-gcp-compliance	Check compliance of GCP configurations to security best practices.	⚠

Microsoft Azure

Azure specific DevSecOps tooling. Tools here cover different areas like inventory management, misconfiguration scanning or IAM roles and policies review.

Name	URL	Description	Meta
Azure Insights	https://github.com/turbot/steampipe-mod-azure-insights	Visualize Azure inventory and permissions through relationship graphs.	⚠

Azure Compliance	https://github.com/turbot/steampipe-mod-azure-compliance	Check compliance of Azure configurations to security best practices.	
PSRule.Rules.Azure	https://github.com/Azure/PSRule.Rules.Azure	Check ARM, Bicep or Live Azure Tenant for security configuration best practices	
PSRule.Rules.AzureDevOps	https://github.com/cloudyspells/PSRule.Rules.AzureDevOps	Check Azure DevOps project for security configuration best practices	

Policy as code

Policy as code is the idea of writing code in a high-level language to manage and automate policies. By representing policies as code in text files, proven software development best practices can be adopted such as version control, automated testing, and automated deployment. (Source: https://docs.hashicorp.com/sentinel/concepts/policy-as-code)

Name	URL	Description	Meta
Open Policy agent	https://github.com/open-policy-agent/opa	General-purpose policy engine that enables unified, context-aware policy enforcement across the entire stack	
Kyverno	https://github.com/kyverno/kyverno	Kyverno is a policy engine designed for Kubernetes	
Inspec	https://github.com/inspec/inspec	Chef InSpec is an open-source testing framework for infrastructure with a human- and machine-readable language for specifying compliance, security and policy requirements.	
Cloud Formation guard	https://github.com/aws-cloudformation/cloudformation-guard	Cloud Formation policy as code	

cnspec	https://github.com/mondoohq/cnspec	cnspec is a cloud-native and powerful Policy as Code engine to assess the security and compliance of your business-critical infrastructure. cnspec finds vulnerabilities and misconfigurations on all systems in your infrastructure including: public and private cloud environments, Kubernetes clusters, containers, container registries, servers and endpoints, SaaS products, infrastructure as code, APIs, and more.	

Chaos engineering

Chaos Engineering is the discipline of experimenting on a system in order to build confidence in the system's capability to withstand turbulent conditions in production.

Reading and manifestos: https://principlesofchaos.org/

Name	URL	Description	Meta

chaos-mesh	https://github.com/chaos-mesh/chaos-mesh	It is a cloud-native Chaos Engineering platform that orchestrates chaos on Kubernetes environments	
Chaos monkey	https://netflix.github.io/chaosmonkey/	Chaos Monkey is responsible for randomly terminating instances in production to ensure that engineers implement their services to be resilient to instance failures.	

Chaos Engine	https://thalesgroup.github.io/chaos-engine/	The Chaos Engine is a tool that is designed to intermittently destroy or degrade application resources running in cloud based infrastructure. These events are designed to occur while the appropriate resources are available to resolve the issue if the platform fails to do so on it's own.	⚠
chaoskube	https://github.com/linki/chaoskube	Test how your system behaves under arbitrary pod failures.	⚠
Kube-Invaders	https://github.com/lucky-sideburn/KubeInvaders	Gamified chaos engineering tool for Kubernetes	⚠
kube-monkey	https://github.com/asobti/kube-monkey	Gamified chaos engineering tool for Kubernetes	⚠

Litmus Chaos	https://litmuschaos.io/	Litmus is an end-to-end chaos engineering platform for cloud native infrastructure and applications. Litmus is designed to orchestrate and analyze chaos in their environments.	⚠
Gremlin	https://github.com/gremlin/gremlin-python	Chaos enginnering SaaS platform with free plan and some open source libraries	⚠
AWS FIS samples	https://github.com/aws-samples/aws-fault-injection-simulator-samples	AWS Fault injection simulator samples	⚠
CloudNuke	https://github.com/gruntwork-io/cloud-nuke	CLI tool to delete all resources in an AWS account	⚠

Infrastructure as code security

Scanning your infrastructure when it is only code helps shift-left the security. Many tools offer in IDE scanning and providing real-time advisory do Cloud engineers.

Name	URL	Description	Meta
KICS	https://github.com/Checkmarx/kics	Checkmarx security testing opensource for IaC	⚠
Checkov	https://github.com/bridgecrewio/checkov	Checkov is a static code analysis tool for infrastructure-as-code	⚠
Trivy	https://github.com/aquasecurity/trivy	Comprehensive scanner for infrastructure-as-code	⚠
terrascan	https://github.com/accurics/terrascan	Terrascan is a static code analyzer for Infrastructure as Code	⚠
cfn_nag	https://github.com/stelligent/cfn_nag	Looks for insecure patterns in CloudFormation	⚠
Sysdig IaC scanner action	https://github.com/sysdiglabs/cloud-iac-scanner-action	Scans your repository with Sysdig IAC Scanner and report the vulnerabilities.	⚠

Terraform Compliance for AWS	https://github.com/turbot/steampipe-mod-terraform-aws-compliance	Check compliance of Terraform configurations to AWS security best practices.	
Terraform Compliance for Azure	https://github.com/turbot/steampipe-mod-terraform-azure-compliance	Check compliance of Terraform configurations to Azure security best practices.	
Terraform Compliance for GCP	https://github.com/turbot/steampipe-mod-terraform-gcp-compliance	Check compliance of Terraform configurations to GCP security best practices.	
Terraform Compliance for OCI	https://github.com/turbot/steampipe-mod-terraform-oci-compliance	Check compliance of Terraform configurations to OCI security best practices.	

Network Intrusion Prevention

Network Intrusion Prevention (NIP) is a security mechanism used to detect and prevent unauthorized access, attacks, or malicious activities on a computer network. It is designed to monitor network traffic in real-time, identify potential threats, and take proactive measures to mitigate them.

Name	URL	Description	Meta

CrowdSec	https://github.com/crowdsecurity/crowdsec)	Crowdsec is an open-source, lightweight software, detecting peers with aggressive behaviours to prevent them from accessing your systems.	

Orchestration

Event driven security help to drive, automate and execute tasks for security processes. The tools here and not dedicated security tools but are helping to automate and orchestrate security tasks or are part of most modern security automation frameworks or tools.

Name	URL	Description	Meta
StackStorm	https://github.com/StackStorm/st2	Platform for integration and automation across services and tools supporting event driven security	
Camunda	https://github.com/camunda/camunda-bpm-platform	Workflow and process automation	

DefectDojo	https://github.com/DefectDojo/django-DefectDojo	Security orchestration and vulnerability management platform	⚠
Faraday	https://github.com/infobyte/faraday	Security suite for Security Orchestration , vulnerability management and centralized information	⚠

Methodologies, whitepapers and architecture

List of resources worth investigating:

- https://dodcio.defense.gov/Portals/0/Documents/DoD%20Enterprise%20DevSecOps%20Reference%20Design%20v1.0_Public%20Release.pdf
- https://dodcio.defense.gov/Portals/0/Documents/Library/DoDEnterpriseDevSecOpsStrategyGuide.pdf
- https://csrc.nist.gov/publications/detail/sp/800-204c/draft
- https://owasp.org/www-project-devsecops-maturity-model/
- https://www.sans.org/posters/cloud-security-devsecops-best-practices/

AWS DevOps whitepapers:

- https://d1.awsstatic.com/whitepapers/aws-development-test-environments.pdf
- https://d1.awsstatic.com/whitepapers/AWS_DevOps.pdf
- https://d1.awsstatic.com/whitepapers/AWS_Blue_Green_Deployments.pdf
- https://d1.awsstatic.com/whitepapers/DevOps/import-windows-server-to-amazon-ec2.pdf
- https://d1.awsstatic.com/whitepapers/DevOps/Jenkins_on_AWS.pdf
- https://d1.awsstatic.com/whitepapers/DevOps/practicing-continuous-integration-continuous-delivery-on-AWS.pdf
- https://d1.awsstatic.com/whitepapers/DevOps/infrastructure-as-code.pdf
- https://d1.awsstatic.com/whitepapers/microservices-on-aws.pdf
- https://d1.awsstatic.com/whitepapers/DevOps/running-containerized-microservices-on-aws.pdf

- https://d1.awsstatic.com/Marketplace/solutions-center/downloads/AppSec-DevSecOps-AWS-SANS-eBook.pdf (AWS + SANS whitepaper)

AWS blog:

- https://aws.amazon.com/blogs/devops/building-end-to-end-aws-devsecops-ci-cd-pipeline-with-open-source-sca-sast-and-dast-tools/
- https://aws.amazon.com/blogs/devops/building-an-end-to-end-kubernetes-based-devsecops-software-factory-on-aws/

Microsoft whitepapers:

- https://azure.microsoft.com/mediahandler/files/resourcefiles/6-tips-to-integrate-security-into-your-devops-practices/DevSecOps_Report_Tips_D6_fm.pdf
- https://docs.microsoft.com/en-us/azure/architecture/solution-ideas/articles/devsecops-in-azure
- https://docs.microsoft.com/en-us/azure/architecture/solution-ideas/articles/devsecops-in-github

GCP whitepapers:

- https://cloud.google.com/architecture/devops/devops-tech-shifting-left-on-security
- https://cloud.google.com/security/overview/whitepaper
- https://services.google.com/fh/files/misc/security_whitepapers_march2018.pdf
- https://cloud.google.com/security/encryption-in-transit/application-layer-transport-security
- https://services.google.com/fh/files/misc/google-cloud-security-foundations-guide.pdf

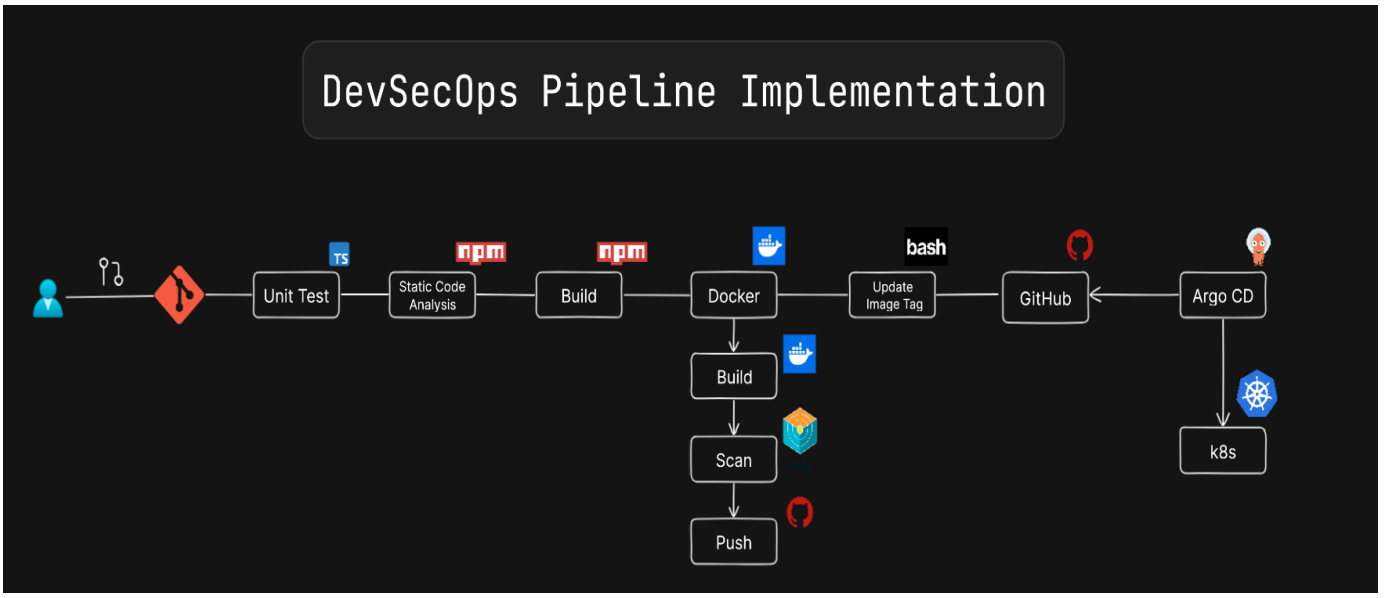
Other

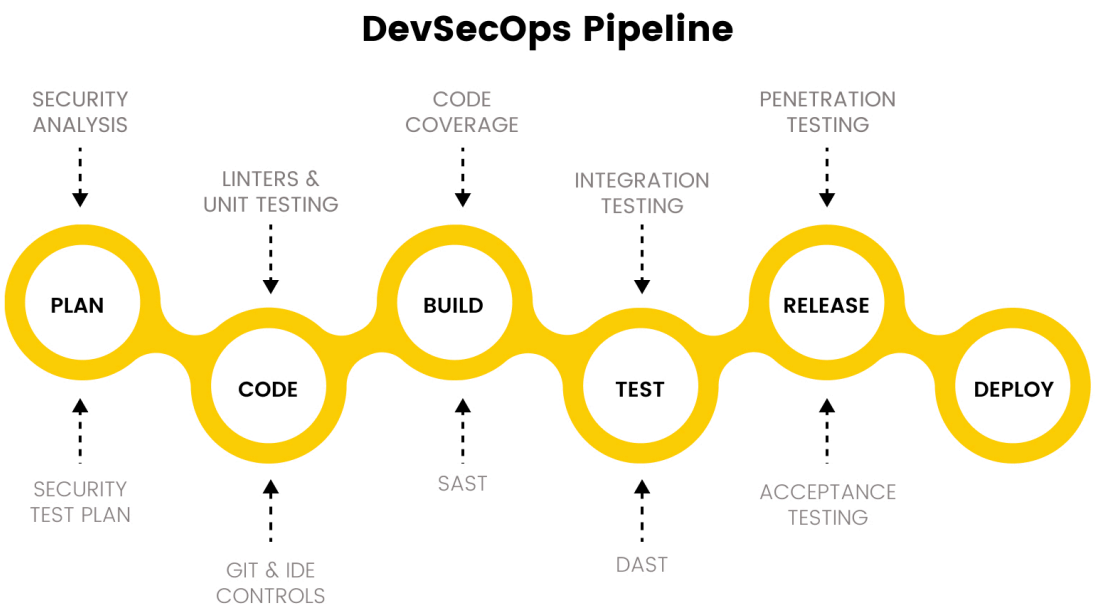
Here are the other links and resources that do not fit in any previous category. They can meet multiple categories in time or help you in your learning.

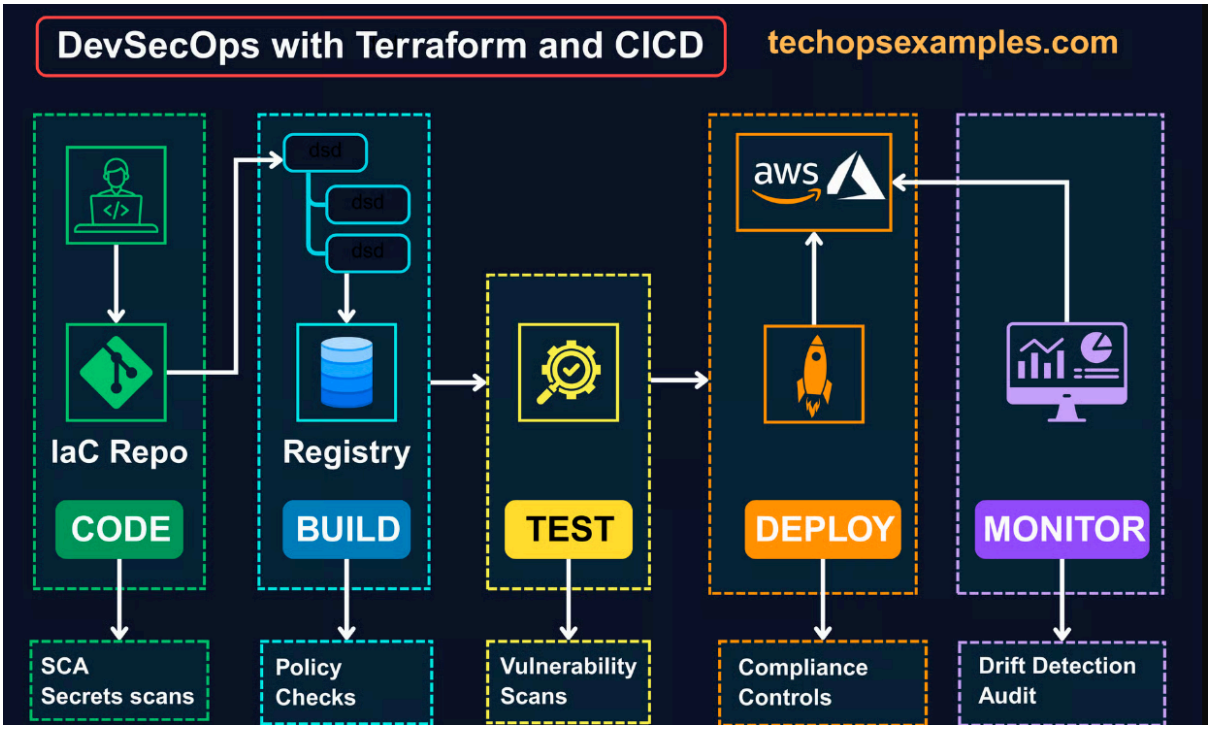
Name	URL	Description	Meta

Automated Security Helper (ASH)	https://github.com/aws-samples/automated-security-helper	ASH is a one stop shop for security scanners, and does not require any installation. It will identify the different frameworks, and download the relevant, up to date tools. ASH is running on isolated Docker containers, keeping the user environment clean, with a single aggregated report. The following frameworks are supported: Git, Python, Javascript, Cloudformation, Terraform and Jupyter Notebooks.	
Mobile security framework	https://github.com/MobSF/Mobile-Security-Framework-MobSF	SAST, DAST and pentesting tool for mobile apps	
Legitify	https://github.com/Legit-Labs/legitify	Detect and remediate misconfigurations and security risks across all your GitHub and GitLab assets	

The DevSec Blueprint	https://devsecblueprint.com	The DevSec Blueprint (DSB) is an a comprehensive, free, and open-source learning guide designed to equip you with the essential skills and knowledge needed to transition into DevSecOps or grow within your DevSecOps career. I explains what you need to know in order to be successful.	




DevSecOps Pipeline
SECURITY ANALYSIS
LINTERS & UNIT TESTING
CODE COVERAGE
INTEGRATION TESTING
PENETRATION TESTING
PLAN
CODE
BUILD
TEST
RELEASE
DEPLOY
SECURITY TEST PLAN
GIT & IDE CONTROLS
SAST
DAST
ACCEPTANCE TESTING


DevSecOps with Terraform and CICD
techopsexamples.com
IaC Repo
CODE
Registry
BUILD
TEST
aws
DEPLOY
MONITOR
SCA
Secrets scans
Policy
Checks
Vulnerability
Scans
Compliance
Controls
Drift Detection
Audit